关于移动端适配你知道多少？

在 web 的世界里，移动端和 PC 的响应式适配其实是两个世界……

1. 视口 viewport

1.1 viewport 基础

viewport 解释为中文就是‘视口’的意思，也就是浏览器中用于显示网页的区域。在 PC 端，其大小也就是浏览器可视区域的大小，所以我们也不会太关注此概念；而在移动端，绝大多数情况下 viewport 都大于浏览器可视区，保证 PC 页面在移动浏览器上面的可视性。为提升可视性体验，针对移动端有了对 viewport 的深入研究。

1.2 viewport 详解

在移动端有三种类型的 viewport: layoutviewport、visualviewport、idealviewport。具体解释如下：

- layoutviewport: 大于实际屏幕， 元素的宽度继承于 layoutviewport，用于保证网站的外观特性与桌面浏览器一样。layoutviewport 到底多宽，每个浏览器不同。iPhone 的 safari 为 980px，通过 document.documentElement.clientWidth 获取。
- visualviewport: 当前显示在屏幕上的页面，即浏览器可视区域的宽度。
- idealviewport: 为浏览器定义的可完美适配移动端的理想 viewport，固定不变，可以认为是设备视口宽度。比如 iphone 7 为 375px, iphone 7p 为 414px。

1.3 viewport 设置

我们通过对几种 viewport 设置可以对网页的展示进行有效的控制，在移动端我们经常会在 head 标签中看到这段代码：

```
<meta name='viewport' content='width=device-width,initial-scale=1,user-scale=no' />
```

通过对 meta 标签三个 viewport 的设置，最终使页面完美展示。下面详细的阐释其具体含义:

- width 设置的是 layoutviewport 的宽度
- initial-scale 设置页面的初始缩放值，并且这个初始缩放值是相对于 idealviewport 缩放的，最终得到的结果不仅会决定 visualviewport，还会影响到 layoutviewport
- user-scalable 是否允许用户进行缩放的设置

对上面的说明通过公式推导进行进一步的解释：

```
// 设定两个变量：
viewport_1 = width;
viewport_2 = idealviewport / initial-scale;

// 则：
layoutviewport = max{viewport_1, viewport_2};
visualviewport = viewport_2;
```

只要 layoutviewport === visualviewport，页面下面不会出现滚动条，默认只是把页面放大或缩小。

1.4 viewport 举例

以下是通过改变 meta viewport 的几个参数的值来算取不同的 viewport：

width	initial-scale	layoutviewport	visualviewport	idealviewport	是否滚动
-	-	980px	980px	375px	否
device-width	1	375px	375px	375px	否
device-width	2	375px	188px	375px	是
device-width	0.5	750px	750px	375px	否
480px	1	480px	375px	375px	是
480px	2	480px	188px	375px	是
480px	0.5	750px	750px	375px	否

以上是针对 iphone 6/7/8 的测试数据，且无论怎么设置 viewport 都具有临界值，即：75 <= layoutviewport <= 10000，75 <= visualviewport <= 1500。

1.5 为什么要设置 viewport

viewport 的设置不会对 PC 页面产生影响，但对于移动页面却很重要。下面我们举例来说明：

1. 媒体查询 @media 响应式布局中，会根据媒体查询功能来适配多端布局，必须对 viewport 进行设置，否则根据查询到的尺寸无法正确匹配视觉宽度而导致布局混乱。如不设置 viewport 参数，多说移动端媒体查询的结果将是 980px 这个节点布局的参数，而非我们通常设置的 768px 范围内的这个布局参数
2. 由于目前多数手机的 dpr 都不再是 1，为了产出高保真页面，我们一般会给出 750px 的设计稿，那么就需要通过设置 viewport 的参数来进行整体换算，而不是在每次设置尺寸时进行长度的换算。

2. 设备像素比 dpr 与 1px 物理像素

2.1 物理像素（physical pixel）

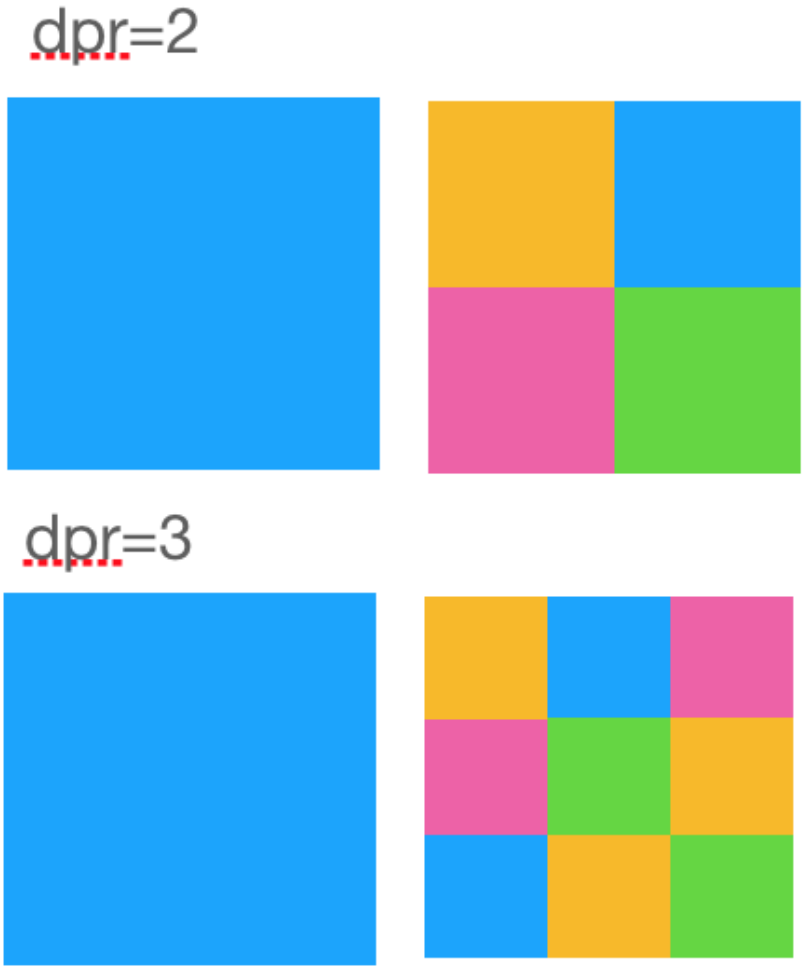
手机屏幕上显示的最小单元，该最小单元具有颜色及亮度的属性可供设置，iphone6、7、8 为：750 * 1334，iphone6+、7+、8+ 为 1242 * 2208

2.2 设备独立像素（density-indenpendent pixel）

此为逻辑像素，计算机设备中的一个点，css 中设置的像素指的就是该像素。老早在没有 retina 屏之前，设备独立像素与物理像素是相等的。

2.3 设备像素比（device pixel ratio）

设备像素比(dpr) = 物理像素/设备独立像素。如 iphone 6、7、8 的 dpr 为 2，那么一个设备独立像素便为 4 个物理像素，因此在 css 上设置的 1px 在其屏幕上占据的是 2 个物理像素，0.5px 对应的才是其所能展示的最小单位。这就是 1px 在 retina 屏上变粗的原因，目前有很多办法来解决这一问题。



2.4 1px 的物理像素的解决方案

从第一部分的讨论可知 viewport 的 initial-scale 具有缩放页面的效果。对于 dpr=2 的屏幕，1px 压缩一半便可与 1px 的设备像素比匹配，这就可以通过将缩放比 initial-scale 设置为 0.5=1/2 而实现。以此类推 dpr=3 的屏幕可以将 initial-scale 设置为 0.33=1/3 来实现。

3. 设备像素比 dpr 与 rem 的适配方案

结合 2、3 部分可以实现 1px 的物理像素这一最小屏幕单位，那在此基础上如可让设计通常提供的 750px 设计稿来完美的适配到多种机型上，使用 rem 是一种解决方式。

3.1 rem 如何设置

rem 是相对于根元素 html 的 font-size 来做计算。通常在页面初始化时加载时通过对 document.documentElement.style.fontSize 设置来实现。

3.2 rem 适配规则

通过对 initial-scale = 1/dpr 的设置，已将对屏幕的描述从物理像素转化到了物理像素上了，这将是后续推导的基础，且设计稿为 750px。

1. 物理像素为 750 = 375 * 2，若屏幕等分为 10 份，那么 1rem = 75px，10rem = 750px;
2. 物理像素为 1125 = 375 * 3，若屏幕等分为 10 份，那么 1rem = 112.5px, 10rem = 1125px;
3. 物理像素为 1242 = 414 * 3，若屏幕等分为 10 份，那么 1rem = 124.2px, 10rem = 1242px;

因此可推导出 rem 的设定方式：

```
 document.documentElement.style.fontSize =
document.documentElement.clientWidth / 10 + 'px';
```

下面我们将 750px 下，1rem 代表的像素值用 baseFont 表示，则在 baseFont = 75 的情况下，是分成 10 等份的。因此可以将上面的公式通用话一些：

```
document.documentElement.style.fontSize =
document.documentElement.clientWidth / ( 750 / 75 ) + 'px';
```

整体设置可参考如下代码：

```
(function (baseFontSize) {
    const _baseFontSize = baseFontSize || 75;
    const ua = navigator.userAgent;
    const matches = ua.match(/Android[\S\s]+AppleWebkit\/(\d{3})/i);
    const isIos = navigator.appVersion.match(/(iphone|ipad|ipod)/gi);
    const dpr = window.devicePixelRatio || 1;
    if (!isIos && !(matches && matches[1] > 534)) {
        // 如果非 iOS, 非 Android4.3 以上, dpr 设为 1;
        dpr = 1;
    }
    const scale = 1 / dpr;
    const metaEl = document.querySelector('meta[name="viewport"]');
    if (!metaEl) {
        metaEl = document.createElement('meta');
```

```
        metaEl.setAttribute('name', 'viewport');
        window.document.head.appendChild(metaEl);
    }
    metaEl.setAttribute('content', 'width=device-width,user-
scalable=no,initial-scale=' + scale + ',maximum-scale=' + scale +
',minimum-scale=' + scale);

    document.documentElement.style.fontSize =
document.documentElement.clientWidth / (750 / _baseFontSize) + 'px';
})();
```

同时为了书写方便可以直接通过 px 布局，然后在打包时利用 pxtorem 库转化为基于 rem 的布局。

4. 视口单位适配方案

将视口宽度 window.innerWidth 和视口高度 window.innerHeight 等分为 100 份，且将这里的视口理解成 idealviewport 更为贴切，并不会随着 viewport 的不同设置而改变。

- vw : 1vw 为视口宽度的 1%
- vh : 1vh 为视口高度的 1%
- vmin : vw 和 vh 中的较小值
- vmax : 选取 vw 和 vh 中的较大值

如果设计稿为 750px，那么 1vw = 7.5px，100vw = 750px。其实设计稿按照设么都没多大关系，最终转化过来的都是相对单位，上面讲的 rem 也是对它的模拟。这里的比例关系也推荐不要自己换算，使用 pxtoviewport 的库就可以帮我们转换。当然每种方案都会有其弊端，这里就不展开讨论。在移动端开发中，理解视口对适配至关重要。

。